**Agile Development Methodology**

**Overview**

[Agile methodology](https://www.uptech.team/blog/agile-methodology-steps) is arguably one of the most popular software development methodologies in recent days. It takes a different approach from the conventional, linear method. Agile focuses on how to satisfy the users instead of emphasizing documentation and rigid procedures.

With Agile, tasks are broken into short sprints that take about 1 to 4 weeks to complete. It’s an iterative model that involves multiple tests as the development progresses. Developers continuously seek [feedback from customers](https://usersnap.com/blog/customer-feedback/)and make changes to the software.

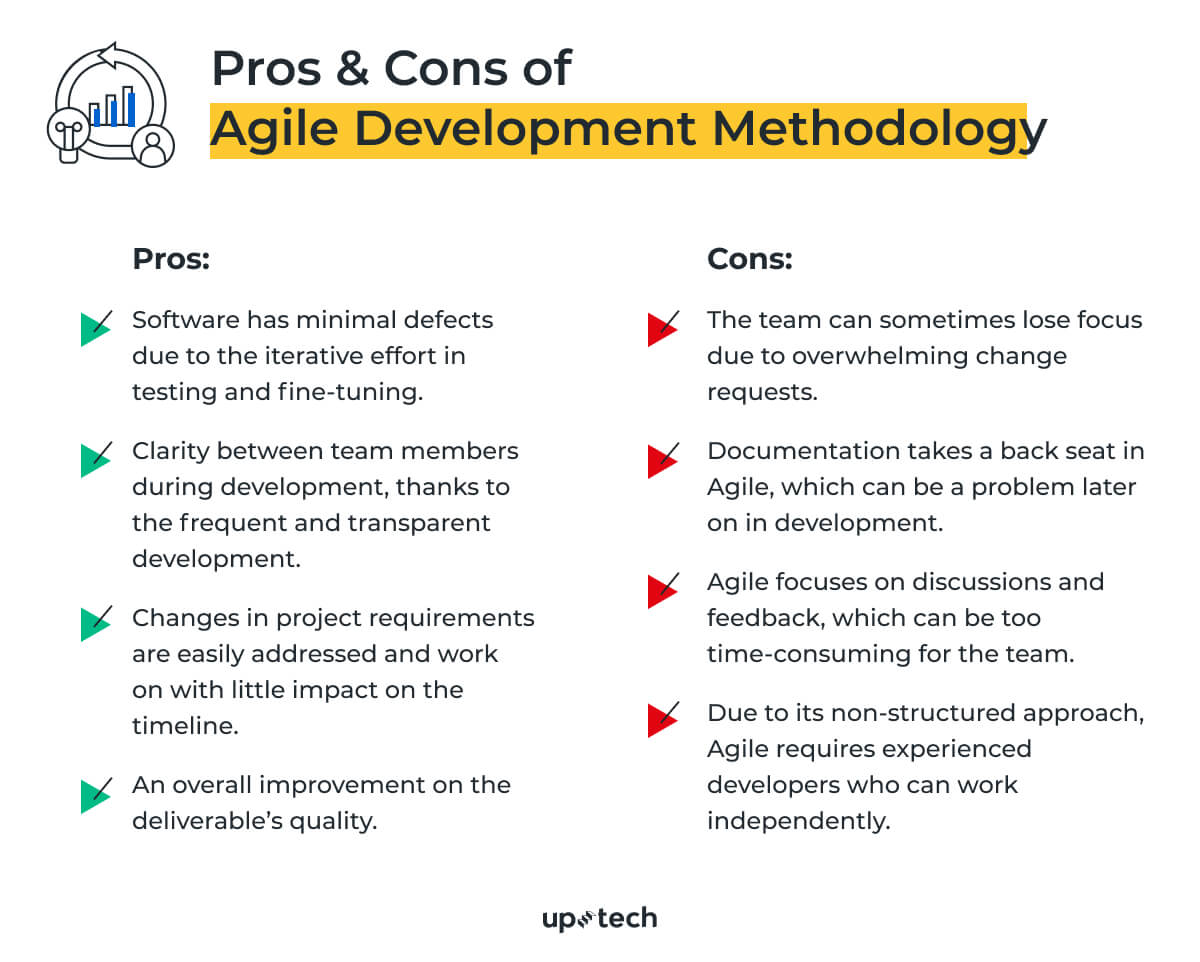
Communication is the priority in Agile, particularly between developers, customers, and users.

**Pros**

* Software has minimal defects due to the iterative effort in testing and fine-tuning.
* Clarity between team members during development, thanks to the frequent and transparent development.
* Changes in project requirements are easily addressed and work on with little impact on the timeline.
* An overall improvement on the deliverable’s quality.

**Cons**

* The team can sometimes lose focus due to overwhelming change requests.
* Documentation takes a back seat in Agile, which can be a problem later on in development.
* Agile focuses on discussions and feedback, which can be too time-consuming for the team.
* Due to its non-structured approach, Agile requires experienced developers who can work independently.



**Suitable For**

The Agile methodology is ideal for projects with fast-changing requirements. If you’re building software in a new niche, you’ll want to use Agile. It works best to implement the additional ideas as you learn more about the market needs. Of course, this assumes that your team of developers is highly-independent and comfortable working in a fast-paced, non-structured environment.

**Waterfall Development Methodology**

**Overview**

Despite decades since it was first used, the waterfall methodology is still relevant in some projects today. It is a simple, linear method where development stages are arranged into sequential, cascading processes.

The waterfall development methodology is easily understood, which makes it popular for teams with lesser design experience. Each stage must be completed before moving on to the next. For example, all the requirements must be established before design can commence.

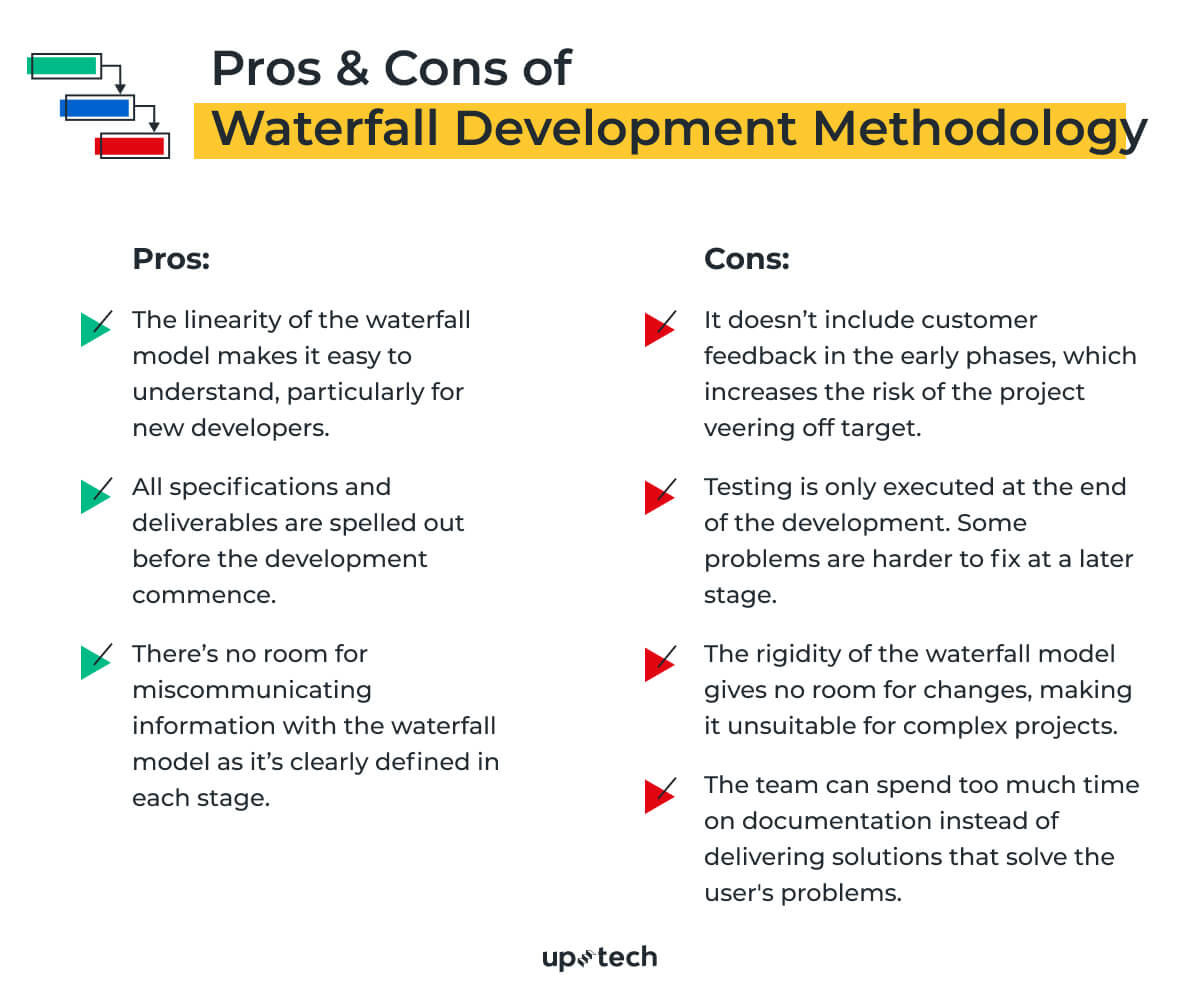
Just like how a waterfall flows in one direction, there’s no going back in this approach. This makes waterfall a non-flexible method and to be avoided for projects with rapidly-changing requirements.

**Pros**

* The linearity of the waterfall model makes it easy to understand, particularly for new developers.
* All specifications and deliverables are spelled out before the development commence.
* There’s no room for miscommunicating information with the waterfall model as it’s clearly defined in each stage.

**Cons**

* It doesn’t include customer feedback in the early phases, which increases the risk of the project veering off target.
* Testing is only executed at the end of the development. Some problems are harder to fix at a later stage.
* The rigidity of the waterfall model gives no room for changes, making it unsuitable for complex projects.
* The team can spend too much time on documentation instead of delivering solutions that solve the user's problems.



**Suitable For**

Use waterfall only when you have a project with clearly-defined scope. It is not suitable for development that involves many unknowns. Waterfall is ideal for projects with predictable outcomes and when you have a team of inexperienced developers.

**Lean Development**

**Overview**

Lean development is born out of the lean manufacturing principles by Toyota. It focuses on minimizing wastage and increasing productivity. With the guiding principles, developers avoid non-productive activities while delivering quality in their tasks.

The Toyota-inspired methodology also emphasizes continuous learning and deferment of decision. It empowers teams to keep an open mind during the course of development and consider all factors before finalizing a decision.

With the lean methodology, developers are tasked to identify bottlenecks that could hamper the process. The goal is to establish an efficient system that works flawlessly. The methodology also emphasizes human respect, which means communication is key to enhancing team collaboration.

[[](https://uptech.team/blog/product-roadmap)](https://uptech.team/blog/product-roadmap" \t "_blank)

‍

**Pros**

* Cuts down wastage in the project, such as redundant codes, unnecessary documentation, and repetitive tasks.
* The overall cost of development is reduced with the lean principles.
* Time-to-market for the software is shortened as lean development promotes efficiency.
* Increased motivation amongst team members as they are empowered with more decision-making authority.

**Cons**

* For lean development to be successful, you’ll need a team of highly-skilled developers, which isn’t easy to put together.
* Less-skilled developers can be overwhelmed by the responsibilities and loss of focus on the project.
* Detailed documentation is needed, which places an enormous burden on the business analyst.



**Suitable For**

Lean development is ideal for a small project with a tight budget to work on. Its principles of minimizing wastage and increasing efficiency mean you’ll be leveraging a small team to achieve tremendous results. Lean development becomes less practical for larger projects, as you’ll need a larger team to take on the tasks.

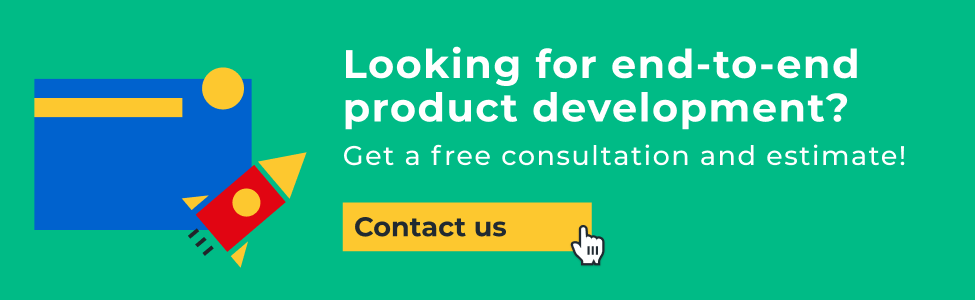
**Prototype Model**

**Overview**

Instead of developing a full-fledged software, the prototype model allows developers to work on the prototype version of the final product. The prototype is then made available for customer testing, evaluation, and feedback.

Based on the gathered feedback, the prototype goes through several iterations of refinement until it’s deemed satisfactory by the customer. The appeal of the prototype approach is its rigorous evaluation that uncovers possible issues before actual development begins.

This approach’s success lies not only in the development team but also in how well they communicate with the customers in carrying out the test. It’s also worth mentioning that the developers often bear the cost of building the prototype.

[[](https://uptech.team/contact)](https://uptech.team/contact" \t "_blank)

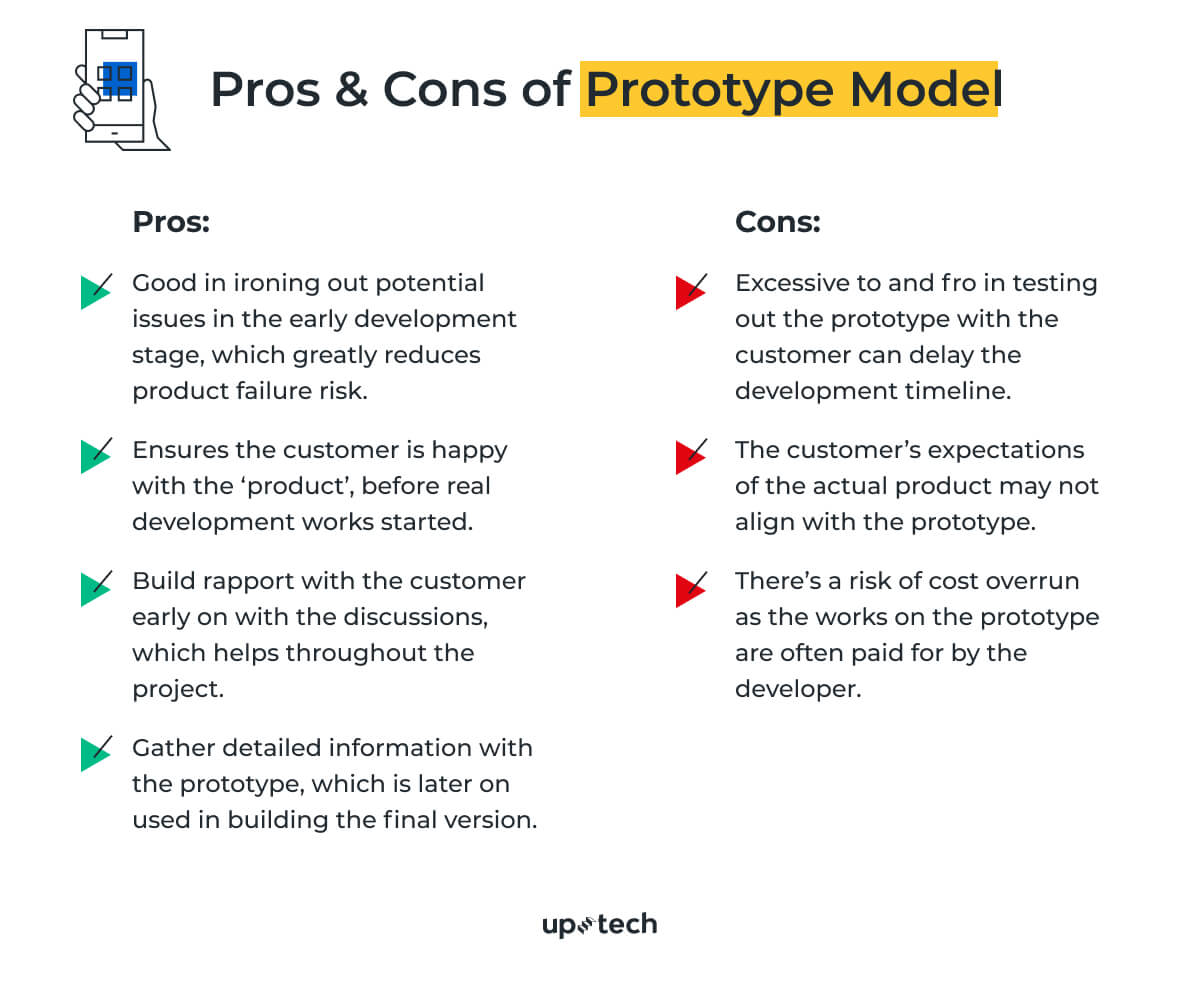
‍

**Pros**

* Good in ironing out potential issues in the early development stage, which greatly reduces product failure risk.
* Ensures the customer is happy with the ‘product’, before real development works started.
* Build rapport with the customer early on with the discussions, which helps throughout the project.
* Gather detailed information with the prototype, which is later on used in building the final version.

**Cons**

* Excessive to and fro in testing out the prototype with the customer can delay the development timeline.
* The customer’s expectations of the actual product may not align with the prototype.
* There’s a risk of cost overrun as the works on the prototype are often paid for by the developer.



**Suitable For**

The prototype model is ideal when you’re building software with many unknowns. For example, an online platform with intense user interaction. With the prototype model, you can find out what works best with users and reduce the risk of developing the actual product.

**Rapid Application Development**

**Overview**

The Rapid Application Development (RAD) model was introduced in 1991 and served as the foundation of modern iterative frameworks. It focuses on getting products built in a much shorter time frame without compromising the quality.

RAD is a 4-step framework, which is defending project requirements, prototyping, testing, and implementation. Unlike linear models, RAD emphasizes building prototypes with the given requirements and testing them out with the customer. This is done through multiple iterations until the customer is happy with the results.

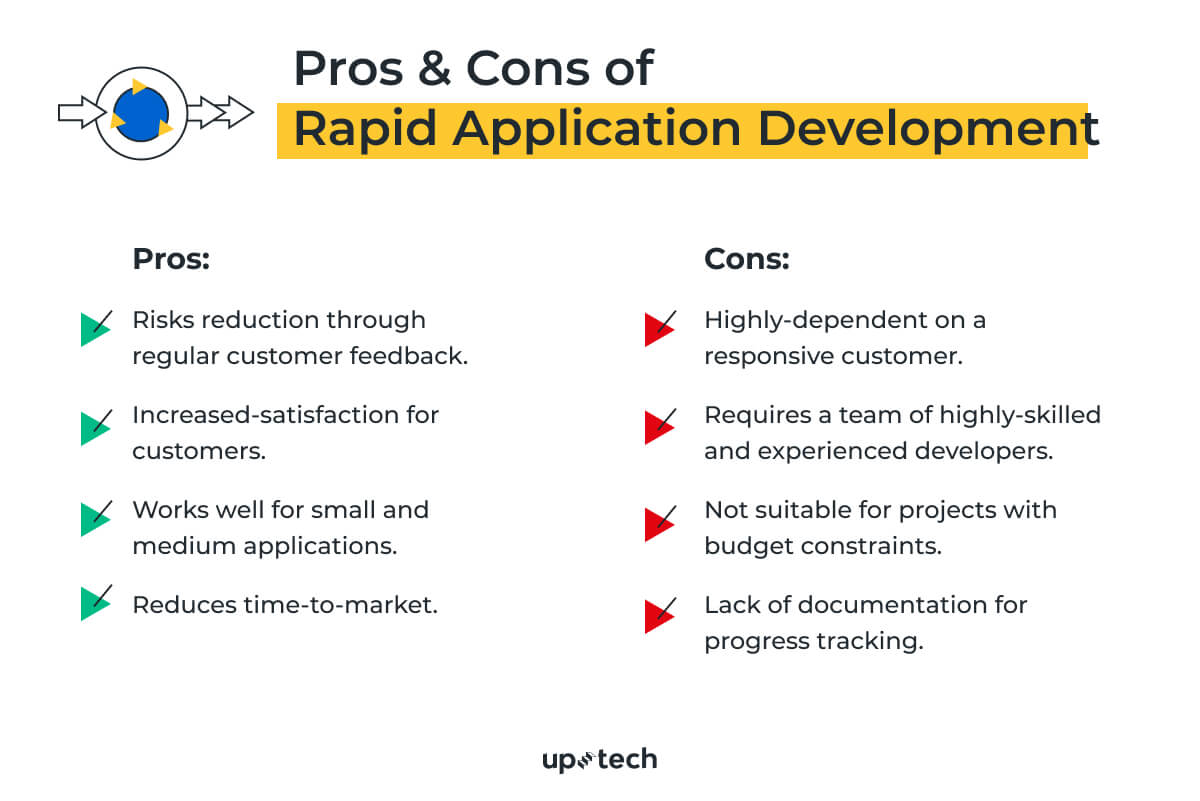
Rigorous testing of the prototype results in valuable feedback, which helps to eliminate much of the product risk. Using RAD leads to high chances of successful product release within the stipulated timeline. RAD often uses development tools that could automate and simplify the development process.

**Pros**

* Risks reduction through regular customer feedback.
* Increased-satisfaction for customers.
* Works well for small and medium applications.
* Reduces time-to-market.

**Cons**

* Highly-dependent on a responsive customer.
* Requires a team of highly-skilled and experienced developers.
* Not suitable for projects with budget constraints.
* Lack of documentation for progress tracking.



**Suitable For**

You’ll get the best results from Rapid Application Development if you have a team of experienced developers and customers who are equally involved in the project. Communication is key to realizing projects with the RAD method. You’ll also need to invest in RAD tools like low-code/no-code applications to speed up the development.

**Dynamic Systems Model**

**Overview**

The Dynamic System Model inherits much of its principles from the Rapid Application Development framework. This refined methodology is aimed at the prompt delivery and aligning project goals to the business needs. It features four iterative phases of feasibility & business study, functional model, design & build, and implementation.

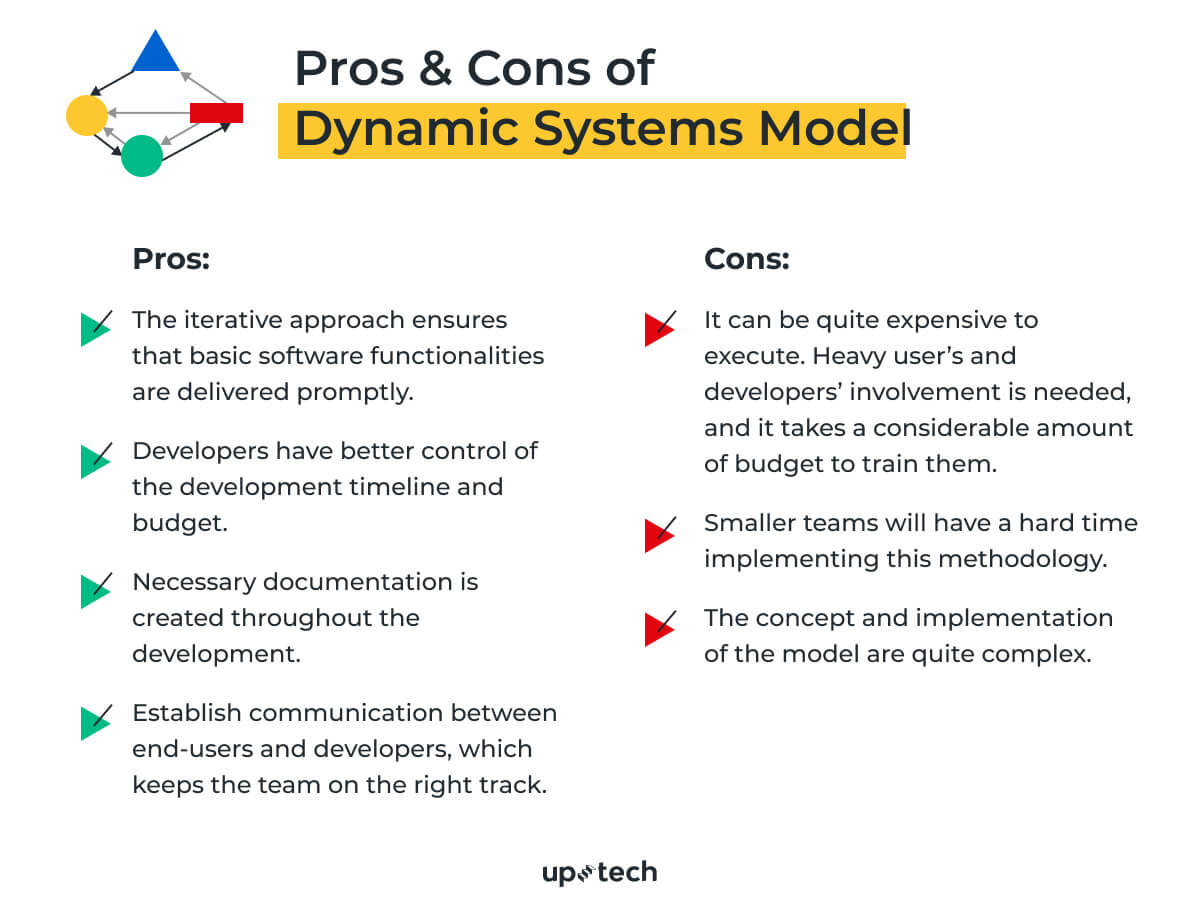
Throughout the process, end users are greatly involved in providing feedback. This reduces the risk of straying from the project goals and requirements. The Dynamic Systems Model also features detailed documentation, which is lacking in most Agile frameworks.

**Pros**

* The iterative approach ensures that basic software functionalities are delivered promptly.
* Developers have better control of the development timeline and budget.
* Necessary documentation is created throughout the development.
* Establish communication between end-users and developers, which keeps the team on the right track.

**Cons**

* It can be quite expensive to execute. Heavy user’s and developers’ involvement is needed, and it takes a considerable amount of budget to train them.
* Smaller teams will have a hard time implementing this methodology.
* The concept and implementation of the model are quite complex.



**Suitable For**

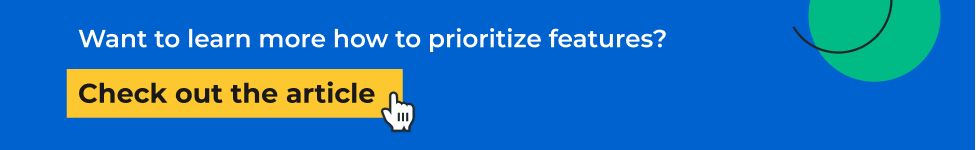
Using the Dynamic Systems Model for small organizations is out of the question. Instead, it is a better fit for large organizations saddled by red-tapes, bottlenecks, and redundancies. It breaks down rigid processes into smaller iterative parts and streamlines communication between different teams.

**Feature Driven Development**

**Overview**

Feature Driven Development, or FDD is a software development methodology based on Agile. Its goal is simple, to prevent confusion that leads to costly rework. FDD is sometimes mistaken as focusing on each of the software features. It is not.

What Feature Driven Development does is breaking down development activities into a feature list on the overall model. For each of the features, developers go through an iteration of planning, designing, and building. Typically a feature should take no longer than two weeks to accomplish.

[[](https://uptech.team/blog/features-prioritization)](https://uptech.team/blog/features-prioritization" \t "_blank)

‍

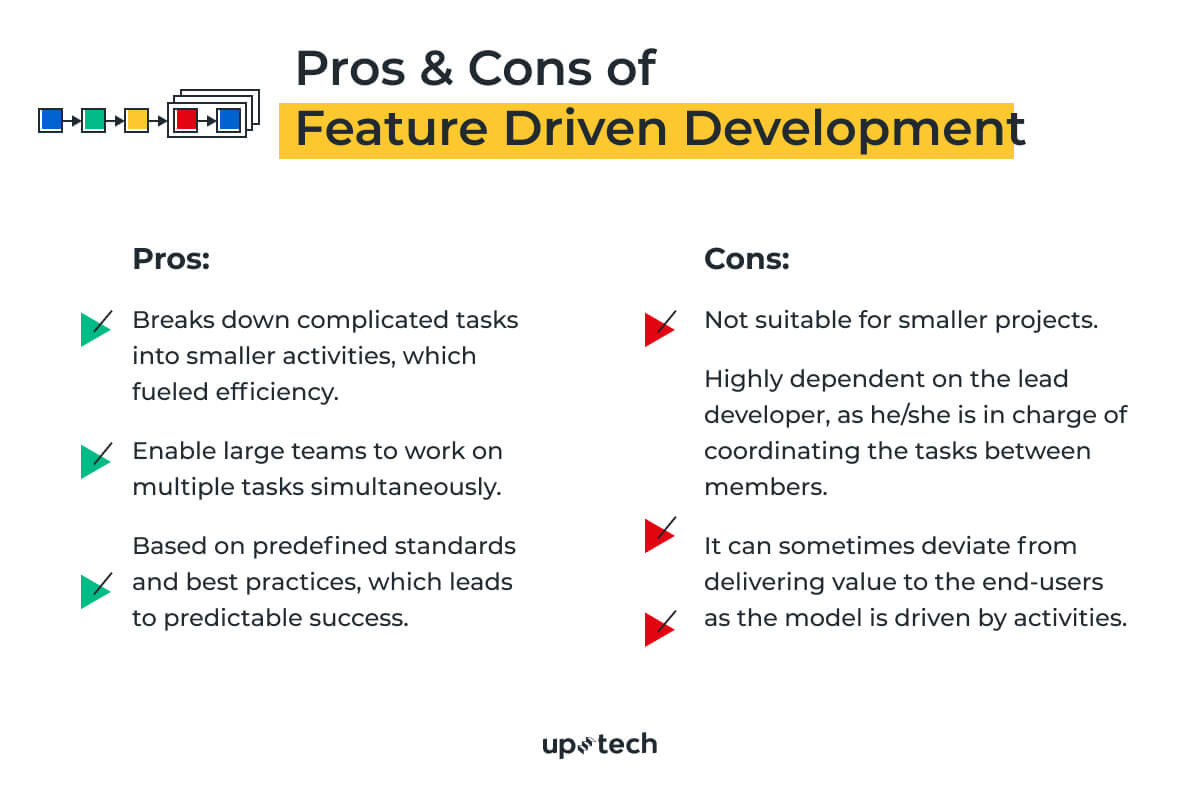
The outcome of FDD is quick, impactful results for each of the activities listed as features. This approach is meant for large teams, and information is communicated through detailed documentation.

**Pros**

* Breaks down complicated tasks into smaller activities, which fueled efficiency.
* Enable large teams to work on multiple tasks simultaneously.
* Based on predefined standards and best practices, which leads to predictable success.

**Cons**

* Not suitable for smaller projects.
* Highly dependent on the lead developer, as he/she is in charge of coordinating the tasks between members.
* It can sometimes deviate from delivering value to the end-users as the model is driven by activities.



**Suitable For**

Feature Driven Development works best for larger teams working on complex projects. It offers the best of an iterative framework but with a more structured approach. Ideally, you’ll want to have a competent lead developer taking charge with FDD.

**Scrum Development**

**Overview**

Scrum is arguably one of the most flexible software development methodologies available. It is based on the Agile philosophy and is favored for its incremental and iterative approaches. The Scrum methodology involves the Product Owner, Scrum Master, and the Development Team.

The product owner takes input from the client and ensures that the team is on track in fulfilling the client’s requirements. Meanwhile, the Scrum Master acts as a facilitator and ensures that team members are familiar with the Scrum process. The team takes charge of executing the development.

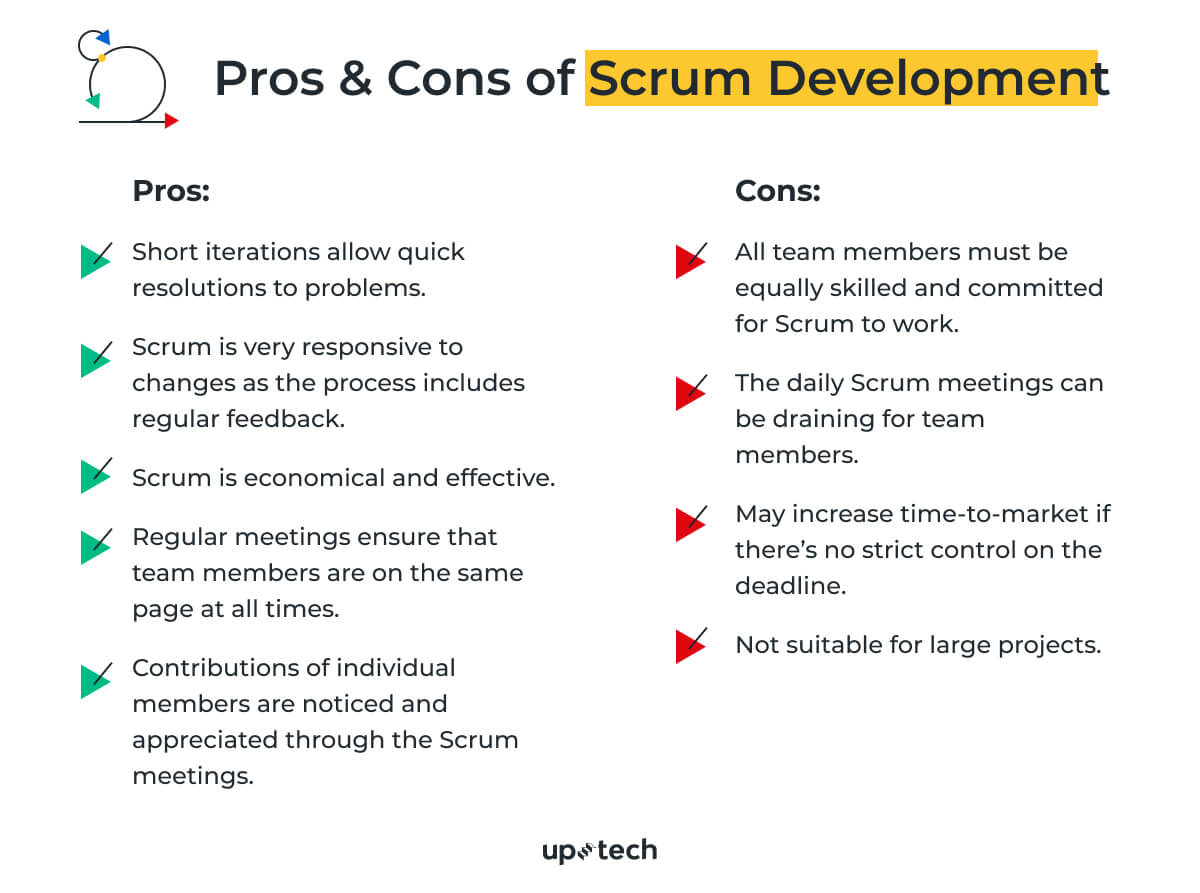
What makes Scrum an ideal methodology in a fast-paced environment is how tasks are executed in sprints. Each sprint takes up to 4 weeks. The speedy execution allows teams to identify issues, introduce solutions, test, and gather feedback in a short period. It makes tackling fast-paced projects much easier.

**Pros**

* Short iterations allow quick resolutions to problems.
* Scrum is very responsive to changes as the process includes regular feedback.
* Scrum is economical and effective.
* Regular meetings ensure that team members are on the same page at all times.
* Contributions of individual members are noticed and appreciated through the Scrum meetings.

**Cons**

* All team members must be equally skilled and committed for Scrum to work.
* The daily Scrum meetings can be draining for team members.
* May increase time-to-market if there’s no strict control on the deadline.
* Not suitable for large projects.



**Suitable For**

Scum is the go-to methodology if you have a project with vague requirements but needs to adapt to frequent changes. For example, you need to get an MVP built quickly and test it out amongst users. Remember that Scrum is only effective if you have a fully-committed and experienced team.

**Summary**

Software development methodologies provide a guided approach to building software and apps. Since the early days of programming, they have been in use and remain pivotal for modern developers..

While new methodologies are introduced throughout the decades, there isn’t one that triumphed over others. Choosing the right software development methodology depends on your team structure, experience, project requirements, goals, budget, and other underlying factors.

We hope you have a better idea of which methodology fits your team best with the detailed descriptions. Else, feel free to get in touch with us for more assistance.